An event-driven optimization framework for dynamic vehicle routing

Victor Pillac, Christelle Guéret, Andrés Medaglia

An event-driven optimization framework for dynamic vehicle routing

Victor Pillac[1,2], Christelle Guéret[*,1], and Andrés L. Medaglia[2]

[1] *École des Mines de Nantes, IRCCyN UMR 6597, Nantes, France*
[2] *Universidad de Los Andes, COPA & CEIBA, Bogotá, Colombia*

June 2011

Technical Report 11/2/AUTO
École des Mines de Nantes
France

Abstract

The real-time operation of a fleet of vehicles introduces challenging optimization problems researches in a wide range of applications, thus, it is appealing to both academia and practitioners in industry. In this work we focus on dynamic vehicle routing problems and present an event-driven framework that can anticipate unknown changes in the problem information. The proposed framework is intrinsically parallelized to take advantage of modern multi-core and multi-threaded computing architectures. It is also designed to be easily embeddable in decision support systems that cope with a wide range of contexts and side constraints. We illustrate the flexibility of the framework by showing how it can be adapted to tackle the dynamic vehicle routing problem with stochastic demands. Computational results show that while our approach is competitive against state-of-the art algorithms, it still ensures greater reactivity and requires less assumptions (e.g., demand distributions).

Keywords: dynamic vehicle routing, event-driven framework, multiple scenario approach, online stochastic optimization, DVRPSD, DVRP

*Corresponding author: gueret@mines-nantes.fr

1 Introduction

The problem of operating a fleet of vehicles arises in many contexts, from pickup and delivery of goods to relocation of trucks in carrier companies. More specifically, Vehicle Routing Problems (VRPs) deal with the design of a set of minimal-cost vehicle routes that serve the demand for goods or services of a group of geographically spread customers, satisfying a group of operational constraints. From an information perspective, such problems generally include two dimensions: *evolution* and *quality* of information [48]. Information evolution relates to the fact that in some problems the information available to the planner may change during the execution of the routes, for example, with the arrival of new customer requests. Information quality reflects possible uncertainty on the available data, for instance, when the demand of a customer is only known as a range estimate of its real demand. In addition, depending on the problem and the available technology, vehicle routes can either be designed a-priori or online. Based on these dimensions, Table 1 identifies four categories of routing problems.

		Information quality	
		Deterministic input	Stochastic input
Information evolution	Input known beforehand	Static and deterministic	Static and stochastic
	Input changes over time	Dynamic and deterministic	Dynamic and stochastic

Table 1: Taxonomy of vehicle routing problems by information evolution and quality.

The *static and deterministic* category includes the classical Vehicle Routing Problem (VRP) as defined by Dantzig [13] in which all information is known beforehand and with certainty. In contrast, problems from the *static and stochastic* class are characterized by input partially known as random variables, which realizations are only revealed during the execution of the routes. Additionally, it is assumed that routes are designed a-priori and only minor changes are allowed afterwards. A common example is the VRP with Stochastic Demands (VRPSD), in which customer demands are uncertain. We refer the interested reader to the surveys by Cordeau et al. [11], Baldacci et al. [2], Laporte [29, 30], and Toth and Vigo [56], for a recent review of these two classes of problems.

In *dynamic and deterministic* problems, also referred to as *online* problems, part or all of the input is unknown and revealed dynamically and unpredictably during the design or execution of the routes. On the other hand, *dynamic and stochastic* problems include partial stochastic knowledge on the dynamically revealed information. For these problems, vehicle routes are redefined in an ongoing fashion, requiring technological support for real time communication between the vehicles and the decision maker (e.g., mobile phones and global positioning systems). Techniques for both classes are reviewed in the studies by Ichoua et al. [25] and Pillac et al. [42].

Dynamism in routing can emerge from different aspects of the problem. The most common source of dynamism is the arrival of new customers with a demand for goods [22, 24, 36] or services [4, 17, 19]. Other researchers consider dynamically revealed demands for a set of known customers [40, 51, 52], dynamic travel times [33, 43, 54], and vehicle availability [31, 32].

Fig. 1 illustrates the Dynamic Vehicle Routing Problem (DVRP), in which new customers appear while the vehicle is executing its route. Before the vehicle leaves the depot (at time t_0), an initial route plans to visit the currently known customers (A, B, C, D, E). While the vehicle executes its route, two new customers (X and Y) appear at time t_1 and the initial route is adjusted to accommodate them. Finally (at time t_f), the executed route is (A, B, C, D, Y, E, X). This example reveals that dynamic routing requires to adjust the routes in an ongoing fashion, which implies real-time communication between vehicles and the dispatching center.

Until recently, the lack or high costs of real-time communication technologies steered vehicle routing research away from dynamic problems [14]. Nevertheless, recent advances in communication and geolocation technologies now allow companies to economically track their fleet in real time. These new technologies lead to the development of Intelligent Transport Systems (ITS),

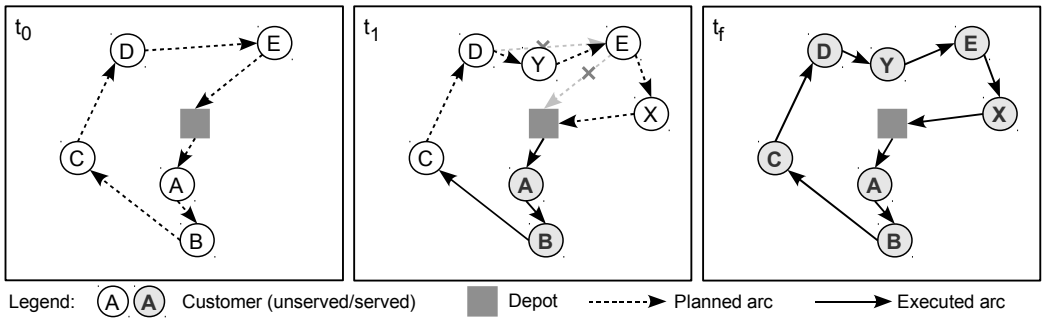

Figure 1: Example of dynamic vehicle routing

and more precisely Advanced Fleet Management Systems (AFMS), that combine hardware and software solutions to provide real time information on the fleet, customers, and road networks.

The development of ITS and AFMS creates new challenges and opportunities for operations research. The advent of these systems demands a new class of efficient optimization algorithms to handle various difficult aspects of fleet management. Nevertheless, Crainic et al. [12] suggest that while the hardware part of ITS has considerably evolved, the corresponding Decision Support Systems (DSSs), and optimization models in particular, have not yet reached their maturity.

From a practical perspective, we can identify the following desirable characteristics of a dynamic routing DSS:

- *Event-driven.* Advances in technology also mean that organizations are now able to react quickly to changes in their environment. Having a DSS which is periodically updated implies longer reaction delays. Thus, a DSS should be driven by the same transactional events that keep the business operating (e.g., customer requests).
- *Parallelized.* As dynamic routing requires fast decisions, the underlying optimization algorithms should be parallelized [12, 25], taking advantage of the now ubiquitous parallel (and distributed) computing architectures able to perform several tasks in parallel.
- *Flexible.* The landscape of vehicle routing problem variants is vast [14]. Thus, a DSS should be easily extensible to account for specific aspects of different applications in a continuously evolving environment.

In this paper, we propose an application-oriented optimization framework for dynamic and stochastic vehicle routing that is event-driven, parallelized and flexible. The rest of this document is organized as follows. Section 2 reviews the literature on dynamic routing optimization techniques and related decision support systems. Section 3 describes the proposed framework, Section 4 illustrates its application to the dynamic VRPSD, and Section 5 presents experimental results. Finally, Section 6 concludes this paper and discusses how the framework can be generalized and extended to other dynamic optimization settings.

2 Literature review

A growing body of research has been carried out on dynamic routing, leading to new optimization techniques and innovative DSSs. In this section we will review some of the most significant contributions in the dynamic routing field.

2.1 Dynamic routing

A wide range of techniques have been developed to address the dynamic nature of routing problems. Dynamic methods can be divided in two categories: *non-anticipative*, which only react to updates in the problem data; and *anticipative*, which take into account knowledge on the dynamically revealed information to anticipate the future. Non-anticipative methods

are designed for dynamic and deterministic problems. They generally are a direct adaptation of static methods such as integer programming [28, 59], large neighborhood search [19], tabu search [4, 17, 23], genetic algorithms [7, 20, 57], and ant colony optimization [16, 39]. Conversely, anticipative methods often make better decisions by using stochastic information available in the form of probability distributions. Anticipative methods are further classified into one of two families: *stochastic modeling* or *sampling*.

Anticipative methods based on stochastic modeling accurately describe the problem's stochasticity. In an early work, Powell et al. [44] formulated the DVRP as a Markov Decision Process (MDP). Nevertheless, the exponential growth of the state and action spaces causes traditional MDPs to stall. This problem has led to the development of Approximate Dynamic Programming (ADP). The main idea behind ADP is to decompose the time in decision epochs. At each decision epoch the goal is to minimize the current deterministic cost plus an approximation of the expected future cost. This technique has been successfully applied to different dynamic fleet management problems [18, 47, 53], freight transport problems [45, 46], and vehicle routing with stochastic demands [40]. The strength of ADP is that it accurately encapsulates stochastic information in the model, but at the expense of a higher complexity and stronger assumptions on the probability distributions.

On the other hand, anticipative methods based on sampling are to some extent simpler, but require more effort to capture the problem's stochasticity. These methods sample the probability distributions to generate *scenarios* that are used to make decisions. Such approaches include the dynamic sample scenario hedge heuristic proposed by Hvattum et al. [21], the tabu search heuristics proposed by Ichoua et al. [24] and Attanasio et al. [1], and the Multiple Scenario Approach (MSA) proposed by Bent and Van Hentenryck [6].

Among the anticipative methods based on sampling, MSA is unique in the sense that it provides a more general framework for dynamic problems. In particular, MSA maintains a *pool of scenarios* with realizations of the problem random variables and a solution to the corresponding deterministic problem. A distinctive feature of MSA is that the next customer to visit is selected based on the whole *scenario pool* by means of a *decision* process. Fig. 2 presents a high level

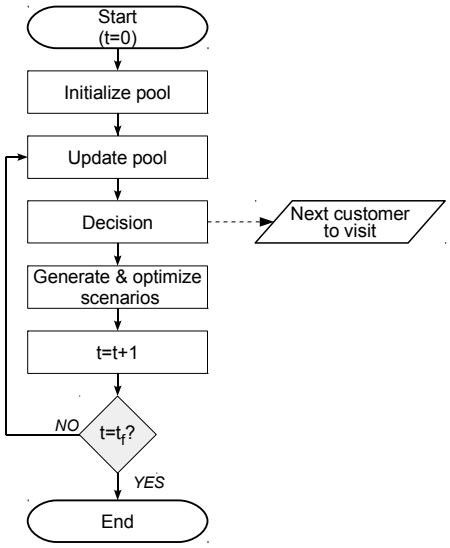

Figure 2: Overview of the original MSA algorithm

flow diagram of MSA. The algorithm starts by initializing the scenario pool based on the currently known information. At time step t, MSA updates the scenario pool to reflect the current environment state, selects the next customer, and optimizes the scenarios. As new information is disclosed, some scenarios might become obsolete and are removed from the pool, leaving space to new ones.

The strength of MSA is that optimization is performed on scenarios and only requires to solve a static and deterministic problem. Therefore this approach is very flexible as it can

virtually be adapted to any problem, provided an optimization algorithm for its static and deterministic version. Nonetheless, its integration in a real-world context is far from trivial, especially considering communication between the method and its environment. Additionally, the fact that it relies on time epochs induces delays between the arrival of new information and its processing.

2.2 Decision support systems for dynamic routing

Zak [60] surveyed a wide range of DSSs for static vehicle routing. With our focus being on dynamic routing, this section reviews the body of research in this area.

The operation of a fleet of vehicles in an urban area is a key component of city logistics [55], and the core subject of various DSSs developments. For instance, Fleischmann et al. [15] presented an event-based DSS that takes into account changing travel times and the arrival of new customers in the context of a local area courier service. The framework continuously optimizes a single routing plan in which new customers are inserted either with an assignment model or insertion algorithms. A similar problem was addressed by Attanasio et al. [1] who showed that the proposed DSS allows for an efficient operation (low administrative cost) as the fleet size (couriers) increases, a key competitive advantage in this sector. Likewise, Barcelo et al. [3] presented a flexible DSS for vehicle routing and scheduling in city logistics and its application to the delivery of goods in two Italian cities. Their DSS includes a real time traffic simulator, connection to common GIS systems, and various routing models and optimization modules. In a different context, Zeimpekis et al. [61] developed a DSS that takes into account unexpected events such as traffic conditions or vehicle breakdowns to re-optimize an existing distribution schedule. Li et al. [31] also studied vehicle breakdowns in an application to waste collection in Brazil. Dynamic DSSs generally rely on specific technology to ensure the communication between vehicles and the dispatching center [61]. In contrast Bieding et al. [8] propose a DSS based on a WAP (Wireless Application Protocol) server and mobile phones to manage the delivery of newspapers.

As pointed out by Crainic et al. [12], there is a gap between state-of-the-art optimization techniques and the optimizers embedded in real-life DSSs. This may be explained by the complexity and level of specialization of certain approaches, that render difficult their extension and integration in an application-oriented context. To address this issue, we propose a flexible optimization framework, based on MSA [58], easily embeddable in any DSS for dynamic routing.

3 Proposed framework

The framework, called jMSA, is a flexible, parallel, and event-driven Java implementation of the multiple scenario approach. The proposed framework has been designed to facilitate and accelerate the development and deployment of MSA-based algorithms embeddable in DSSs. This section presents the proposed framework in detail.

3.1 Scenarios and decisions

Scenarios capture uncertainty in MSA. Each scenario contains a realization of the random variables, and a solution to the static and deterministic problem defined by this realization. For instance, in the Dynamic VRPSD (DVRPSD), in which vehicles can be dynamically rerouted, each scenario contains a realization of the customer demands; while in the DVRP, it contains a set of sampled (potential) customers, aside from the known customers. An optimization algorithm is used to solve the static and deterministic routing problem defined by both actual and sampled data. Fig. 3 illustrates how scenarios are generated for the DVRP. Solely based on the actual customers, the optimal tour would be (A, B, E, D, C), which ignores two zones (gray areas) where customers are likely to appear. By sampling the customer spatial distributions, customers X, Y and Z are generated, and the new optimal tour is (C, X, Y, B, A, Z, E, D). Removing the sampled (potential) customers leads to the tour (C, B, A, E, D) which is suboptimal based on a myopic cost evaluation, but leaves room to accommodate new customers at a lower cost.

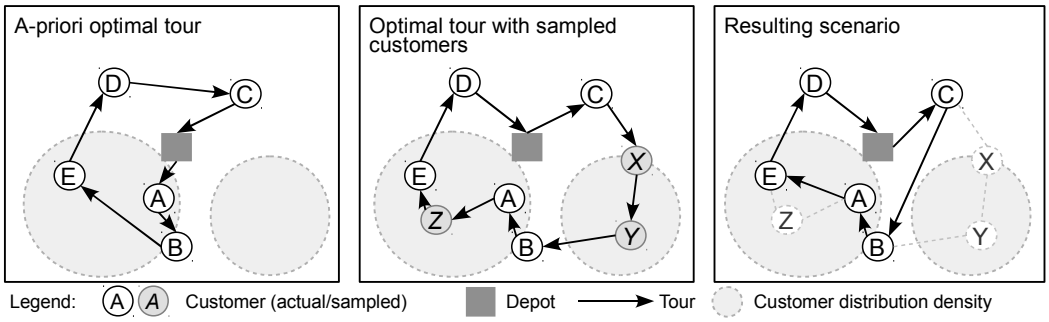

Figure 3: Scenario generation in MSA

Another key element in MSA is the *decision process*, which defines how to select the next customer to serve based on the information of the scenario pool. MSA's accuracy relies to a great extent on the decision process, being the most common algorithms *expectation*, *consensus*, and *regret*. The *expectation* algorithm [9] evaluates the cost of visiting each customer first, by forcing its visit and reoptimizing each scenario. The *consensus* algorithm [6] selects the customer appearing first with the highest frequency. Finally, the *regret* algorithm [5] approximates the cost of visiting each customer first.

The jMSA framework unifies these decision processes in the generic Algorithm 1, in which a subset of candidate customers (line 1) is evaluated against the scenario pool (line 6) to select the *best* one (line 9). The evaluation of each customer reflects how desirable it is to serve it first depending on the objective. In most routing problems, the customer with the highest evaluation should be the one that ensures the lowest expected routing distance when visited first.

Algorithm 1 A general algorithm for the decision process in jMSA

Input: scenario pool $\mathcal{P}$, set of pending customers $\mathcal{R}$
Output: r^* the next customer to serve

```
1:  C ← selectCandidates(R, P)
2:  f* ← −∞, r* ← ∅
3:  for all r ∈ C do
4:      f ← 0
5:      for all s ∈ P do
6:          f ← f + evaluateRequestProfit(r, s)
7:      end for
8:      if f > f* then
9:          f* ← f, r* ← r
10:     end if
11: end for
12: return r*
```

3.2 Event-driven interaction

The original description of MSA is implicitly based on the notion of *time steps* introduced by Kilby et al. [27]. By design, this time discretization in intervals implies a time lag between an update in the problem data, such as the arrival of a new customer, and the response of the system, corresponding to the time before the next time step. Consequently, in jMSA we propose a description of MSA from an event-driven perspective, suitable for its integration as a component of a real-world decision support system.

Fig. 4 illustrates a typical sequence of events while routing a single vehicle in a dynamic context. The *environment* refers to the *real-world*, the *DSS* is assumed to be based on the MSA algorithm, and *active* (*idle*) times are represented with a continuous (dotted) segment. While the vehicle is parked at the depot, the MSA procedure initializes a scenario pool based on the

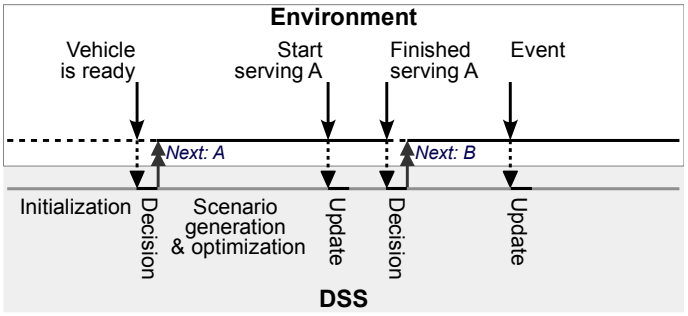

Figure 4: Time line of events for a the dynamic routing of a single vehicle

currently known customers. Once the vehicle is ready (first dotted arrow), MSA analyzes the scenario pool and instructs the vehicle to service customer A (first double-headed arrow). While the vehicle is traveling towards customer A, MSA generates and reoptimizes the scenario pool. When the vehicle reaches its destination, an event is sent to the system (second dotted arrow) and triggers an update of the scenario pool. The remaining service time is used by MSA to reoptimize the pool until the vehicle is ready to depart. This event (third dotted arrow) triggers the decision procedure, which recommends visiting customer B (second double-headed arrow). At some point in time while the vehicle is traveling to B, an event (last dotted arrow) triggers an update of the scenario pool. Such event could be the arrival of a new customer in the DVRP, or an update in the traffic information in the case of routing with dynamic travel times.

The main advantage of this event-driven interaction between the environment and the system is that it increases the responsiveness of the DSS by feeding real-time information to the system and communicating decisions without delay.

3.3 Framework design

As illustrated in Fig. 5, the proposed framework is divided in two layers: a *kernel*, common to all dynamic combinatorial optimization problems; and a *problem layer*, with problem-specific components.

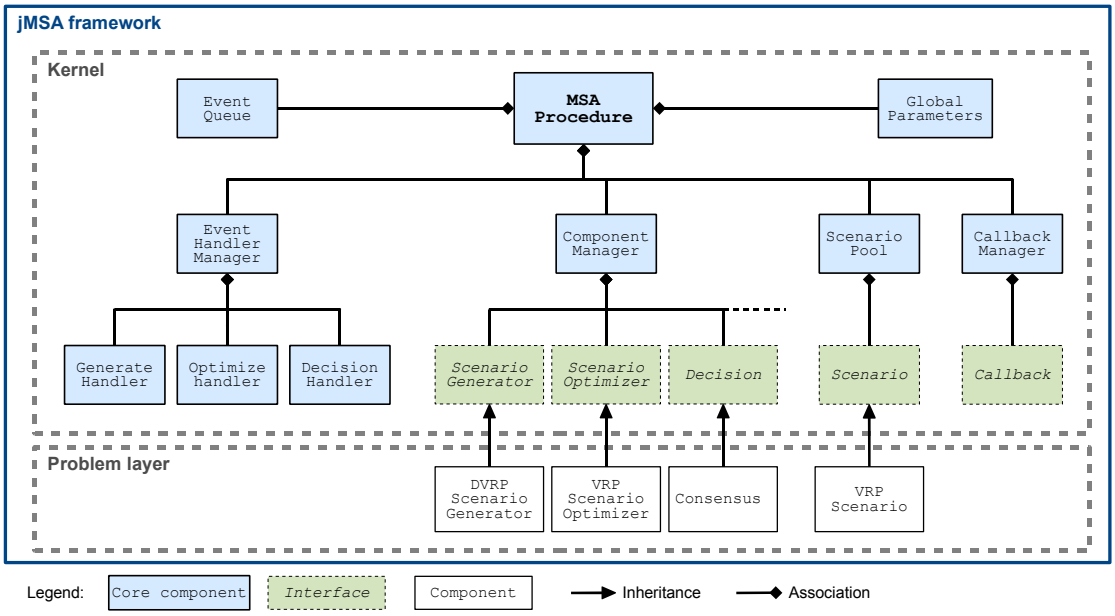

Figure 5: Design overview of the jMSA framework

The central component of the *kernel* is the `MSAProcedure`, which contains the logic of the algorithm and instantiates all other components. The `MSAProcedure` is configured via the `GlobalParameters` that can be set programmatically or via a configuration file.

The event-driven behavior is modeled using two elements: *events* and *event handlers*. Fig. 6 shows how events drive the framework. The MSA procedure continuously dequeues events from the *event queue*, and then processes them by using the corresponding *event handler* in the *event handler manager*.

Events are designed to increase the framework responsiveness. To ensure that *important* events are handled first, events are prioritized and the event queue is sorted accordingly. Additionally, some events are *preemptive*, meaning that handling of a non-preemptive event is always aborted in favor of a preemptive event.

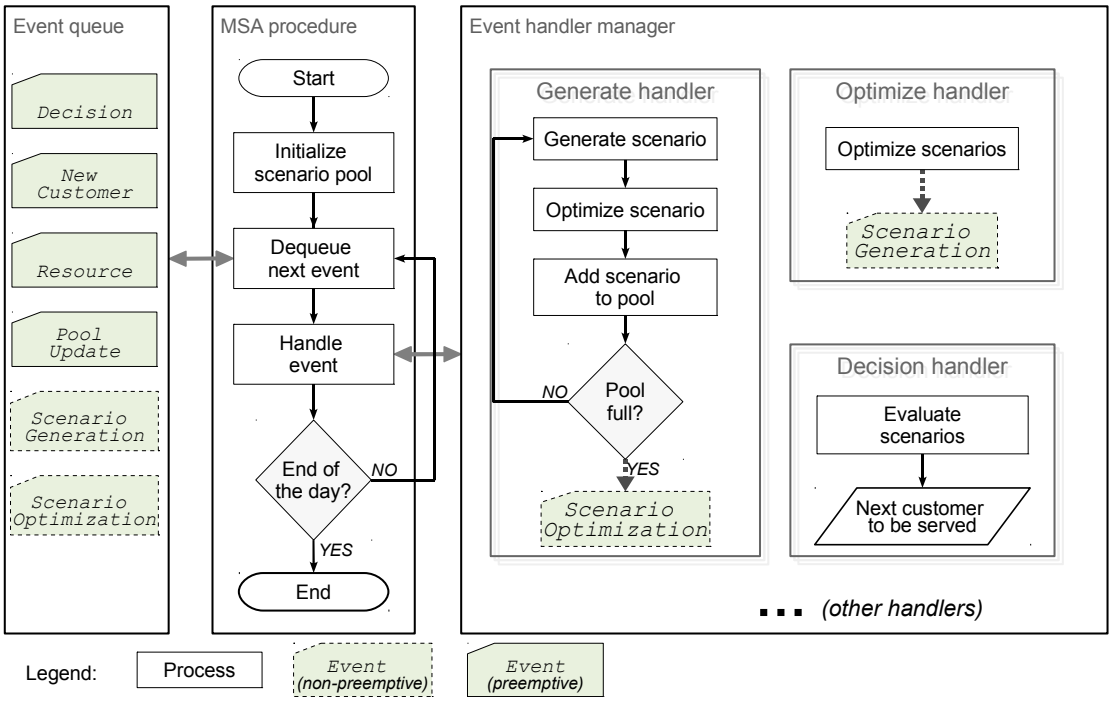

Figure 6: Event-driven MSA framework

Event handlers define at a very high level what actions are triggered by a given event. By design, these handlers do not contain any problem-specific logic which is rather delegated to *components*. The *component manager* contains references to all components and acts as an interface between event handlers and problem-specific implementations.

Fig. 7 illustrates how event handlers and components interact by means of the `ScenarioGeneration` event. First, `GenerateHandler` calls the `generateScenario` method of the `ComponentManager` that internally uses the registered `ScenarioGenerator`. Then it calls the `optimizeScenario` method, delegated to the instance of `ScenarioOptimizer` in use, and adds the scenario to the pool. The process repeats until the pool is full, moment when the event handling terminates by raising a `ScenarioOptimization` event that is further pushed to the event queue.

The framework includes a *callback* system that provides users with further control over the MSA procedure. Users may implement a callback simply by extending the generic `Callback` provided in the framework, and registering it in the MSA procedure. User-defined callbacks are automatically invoked at specific points of the procedure and allow customized uses such as logging to a file or dynamic parameter tunning.

Tied, yet decoupled to the kernel, the jMSA framework offers a problem-specific layer containing components that provide ready-to-use functionalities for common dynamic combinatorial

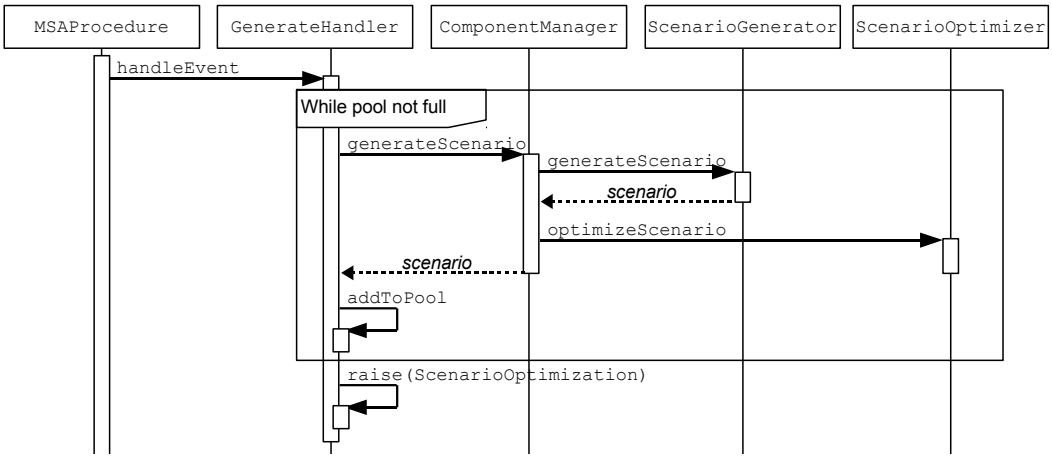

Figure 7: Interaction between the `GenerateHandler` and the different components.

optimization problems. Fig. 5 illustrates some components that could be combined for the DVRP. `Consensus` is an implementation of the consensus algorithm that is common to many dynamic problems solved under MSA; `VRPScenario` is an implementation of `Scenario` for routing problems containing a set of routes; `VRPScenarioOptimizer` is a generic solver for the VRP; and finally `DVRPScenarioGenerator` is the only component specific to the DVRP that is responsible for the generation of new scenarios.

This two-layer architecture ensures flexibility and extensibility. While kernel elements are defined at a high level and are designed to be problem independent, the problem layer provides implementations for specific problems. Thus, users only have to define or extend components, in particular for scenario generation and optimization, without worrying how they will be integrated in the MSA procedure.

3.4 Parallelization via multi-threading

The ubiquitous presence of multi-core processors can be exploited in parallelizable algorithms such as MSA. Nevertheless, parallelization often comes at the price of a higher (implementation) complexity. The jMSA framework offers multi-threaded parallelization of the most time-consuming tasks, hiding it from the user. That is, under jMSA, users do not have to explicitly write a parallel algorithm, but simply rely on the `ComponentManager` which internally distributes tasks among different threads.

Fig. 8 illustrates how threads interact within the jMSA framework. At time t_0 the *MSA thread* dequeues an `OptimizePool` event, and processes it with the corresponding `OptimizeHandler`. In parallel to the MSA thread, two other threads are started by the `ComponentManager` to optimize the scenarios of the pool. At t_1, a preemptive `NewCustomer` event is pushed by the environment, causing the MSA thread to prematurely abort the optimization. To avoid inconsistencies, the main thread waits for the pool executor to terminate, sends a signal to the callback thread to notify that the `OptimizePool` event was handled, and raises a `GenerateScenarios` event. Finally, the procedure dequeues the `NewCustomer` event, which has a higher priority than the `Decision` event, and processes it.

It is worth noting that aside from time-consuming tasks such as scenario generation and optimization, parallelization is also used to execute callbacks. Callbacks can be particularly useful when writing files or updating the state of a user interface as it does not affect the performance of the main algorithm. This behavior can be overridden using synchronous callbacks.

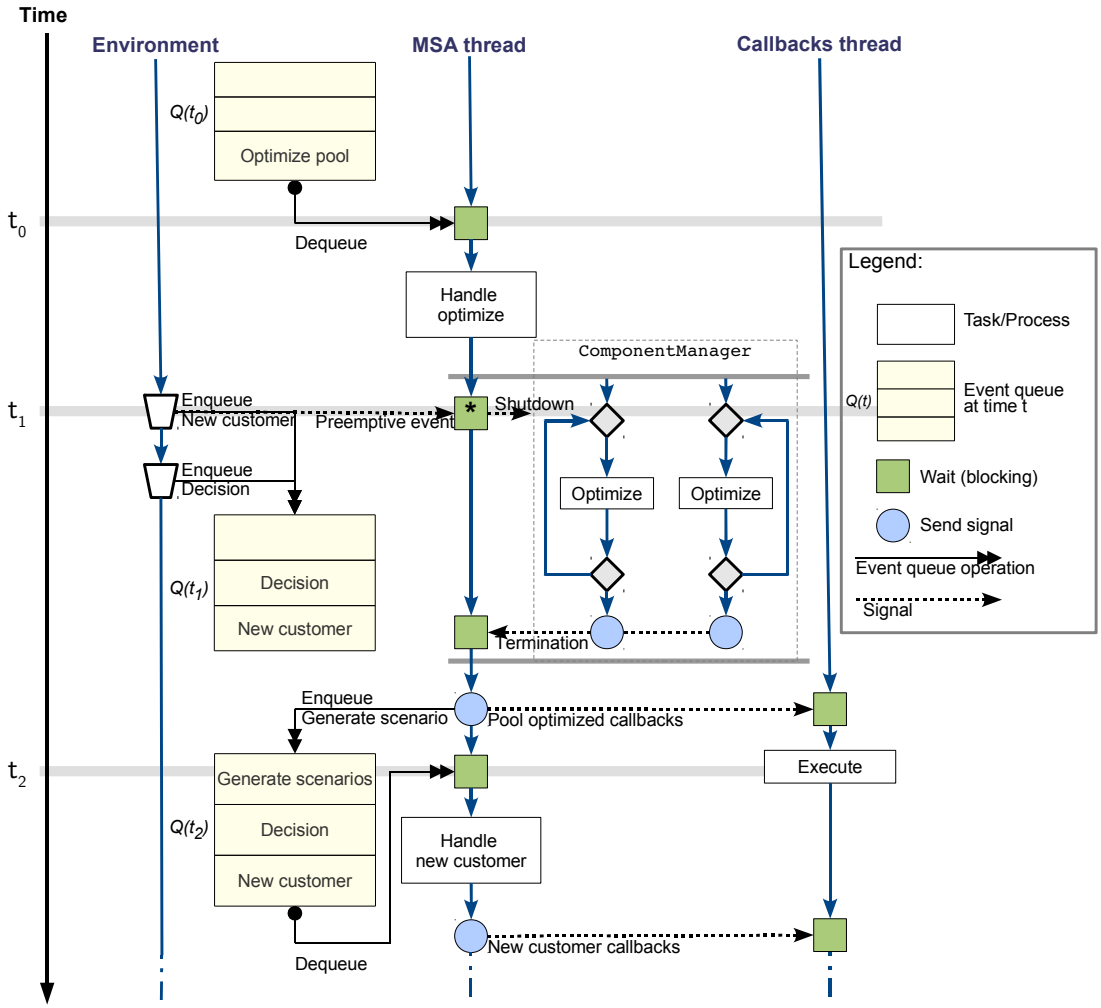

Figure 8: Multiple threads interacting in jMSA

4 Application to the dynamic VRP with stochastic demands (DVRPSD)

This section illustrates the flexibility of the jMSA framework on the dynamic VRP with Stochastic Demands (DVRPSD).

4.1 Problem description

The VRPSD consists in designing a-priori a minimal expected-cost routing plan composed of K routes, such that each customer is visited exactly once. The fundamental difference between the classic VRP and the VRPSD is that in the latter customer demands are known as random variables. The randomness in the VRPSD implies that a customer demand realization might exceed the vehicle remaining capacity, leading to a route *failure* that requires a *recourse action*. An intuitive recourse action is for the vehicle to go back to the depot to restore its initial capacity and then resume its route [34], or to allow the service of additional customers before returning to the depot [41]. It is important to stress that in this context all customers are known beforehand and the only dynamically revealed information is the realization of the customer demands.

Uncertainty in the VRPSD has been addressed by various solution approaches, of which the two most studied are the *Chance Constrained Programming (CCP)* and the *Stochastic Program-*

ming with Recourse (SPR). Both methods are based on a two-stage approach: the first phase, builds *robust* routing plan; while the second phase takes place, takes recourse (corrective) actions as the realizations of the customer demands are unveiled. The conceptual difference between the two approaches lies in the objective of the first-stage optimization: in CCP, the goal is to ensure an upper bound on the probability of a failure, regardless of the expected cost of the second phase; while SPR seeks the minimization of the total expected cost, including recourse actions.

Henceforth, we focus on the single-vehicle Dynamic VRPSD (DVRPSD) ($K = 1$), where it is possible to reroute the vehicle upon new demand realizations, allowing more complex recourse actions. Literature on the DVRPSD is scarce, with the main contributions being the work by Novoa and Storer [40], Secomandi [51], and Secomandi and Margot [52]. All of them consider the case where customer demands are discrete and uniformly distributed. We now show how the jMSA framework can be adapted to tackle this problem. Also, we illustrate how under the proposed approach we can easily relax the assumptions on the demand distributions, thus leading us to the solution of a more general problem with jMSA.

4.2 Scenarios and decisions

In the context of the DVRPSD, the only unknown data is the customer demand realization. Thus, scenarios contain different realizations of the customer demands, along with a feasible routing for these values. As the vehicle can go back to the depot during its service, a scenario can contain different routes that will be executed in a sequential order by the same vehicle.

The fact that customer locations are identical across scenarios suggests that different scenarios might have similar routes. Thus, we decided to use the consensus algorithm to select the next customer to visit. Let us consider the scenario pool of Fig. 9. The customers who have already been served (4 and 1) appear first in all scenarios, while customers 2, 3, 5, and 6, appear in varying order depending on the scenario sampled demands. Considering that customer 2 appears first in 2 out of 4 scenarios, by consensus it is selected as the next customer to visit. As shown in Algorithm 1, the function `selectCandidates` (line 1) returns the set of unserved customers while `evaluateRequestProfit` (line 6) returns 1 if customer r appears first in the scenario; 0, otherwise.

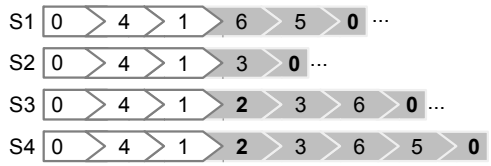

Figure 9: Example of the decision process by consensus in a 4-scenario pool. Each scenario contains customers who have been visited (in white) and customers yet to be visited (in gray).

4.3 Optimization

To optimize scenarios we use an Adaptive Variable Neighborhood Search (AVNS), which is an extension of the Variable Neighborhood Search (VNS) [38]. The main difference between AVNS and VNS is that neighborhoods are not explored sequentially, but randomly selected weighted by their previous *performance*. Our implementation uses an average ratio of the improvement to time as a metric of neighborhood performance, and maintains this information between calls of the optimization procedure. Neighborhoods with a better performance are more likely to be explored first, leading to a self-tuning algorithm. Our MSA scheme benefits from this automatic self-tuning behavior as the optimization procedure is called numerous times on similar instances (i.e., scenarios).

Algorithm 2 presents an outline of the AVNS algorithm. The algorithm initializes with the whole set of neighborhood structures (line 2), then it selects a neighborhood (line 4) to randomly perturb the current solution (line 5), and improves it by applying a local search procedure (line 6). If the new solution is accepted (line 8), then the new solution becomes the current solution

(line 9), and the set of active neighborhood structures is reset (line 10). Otherwise, the current neighborhood is removed from the set of active neighborhoods (line 12). At each iteration, the performance of the current neighborhood is updated (line 7). This process iterates until all neighborhoods have been explored with no improvement.

Algorithm 2 The Adaptive Variable Neighborhood Search algorithm

Input: feasible solution $\mathbf{x}$, evaluation function z , and set of neighborhood structures $\mathcal{N} = \{N_1, .., N_K\}$
Output: best solution found $\mathbf{x}^*$

1: $\mathbf{x}^* \leftarrow \mathbf{x}$
2: $\mathcal{N}_c \leftarrow \mathcal{N}$
3: **while** $\mathcal{N}_c \neq \emptyset$ **do**
4: $N \leftarrow$ `selectNeighborhood`$(\mathcal{N}_c)$
5: $\mathbf{x}' \leftarrow$ `shake`$(N, \mathbf{x})$
6: $\mathbf{x}' \leftarrow$ `localSearch`$(\mathbf{x}')$
7: `updatePerformance`$(\mathcal{N}, \mathbf{x}, \mathbf{x}')$
8: **if** `accept`$(\mathbf{x}', \mathbf{x})$ **then**
9: $\mathbf{x} \leftarrow \mathbf{x}'$
10: $\mathcal{N}_c \leftarrow \mathcal{N}$
11: **else**
12: $\mathcal{N}_c \leftarrow \mathcal{N}_c \setminus \{N\}$
13: **end if**
14: **if** $z(\mathbf{x}') < z(\mathbf{x}^*)$ **then**
15: $\mathbf{x}^* \leftarrow \mathbf{x}'$
16: **end if**
17: **end while**
18: **return** $\mathbf{x}^*$

In our experiments we used the two neighborhoods structures *Or-opt* and *string-exchange* for the perturbation, and a Variable Neighborhood Descent (VND) based on *swap* and *2-opt* as local search (line 6). A more detailed description of these neighborhoods can be found in the paper by Irnich et al. [26]. The initial solution is obtained by a Clarke and Wright (CW) heuristic [10] in which the saving list is randomized, as presented in Mendoza et al. [35], leading to the CW+AVNS algorithm.

4.4 Failure handling

A route fails when a customer demand exceeds the vehicle's remaining capacity. Thus, the MSA procedure becomes aware of a route failure as soon as a `Resource` event is raised upon the arrival at the customer location. As a consequence, the route failures handling must be defined at the event handler level, by checking if the demand of the current customer is larger than the vehicle remaining capacity, and updating the scenario pool accordingly.

4.5 User interface

To illustrate the use of callbacks we developed a user interface shown in Fig. 10. The main panel (right) presents in real time the unserved (white) and served (dark gray) customers, the vehicle destination (light gray), and the executed route (arrows). The left panel, displays a log of events of jMSA and echoes the configuration settings. By means of a callback registered in the MSA procedure, all the information in the interface is updated in real time.

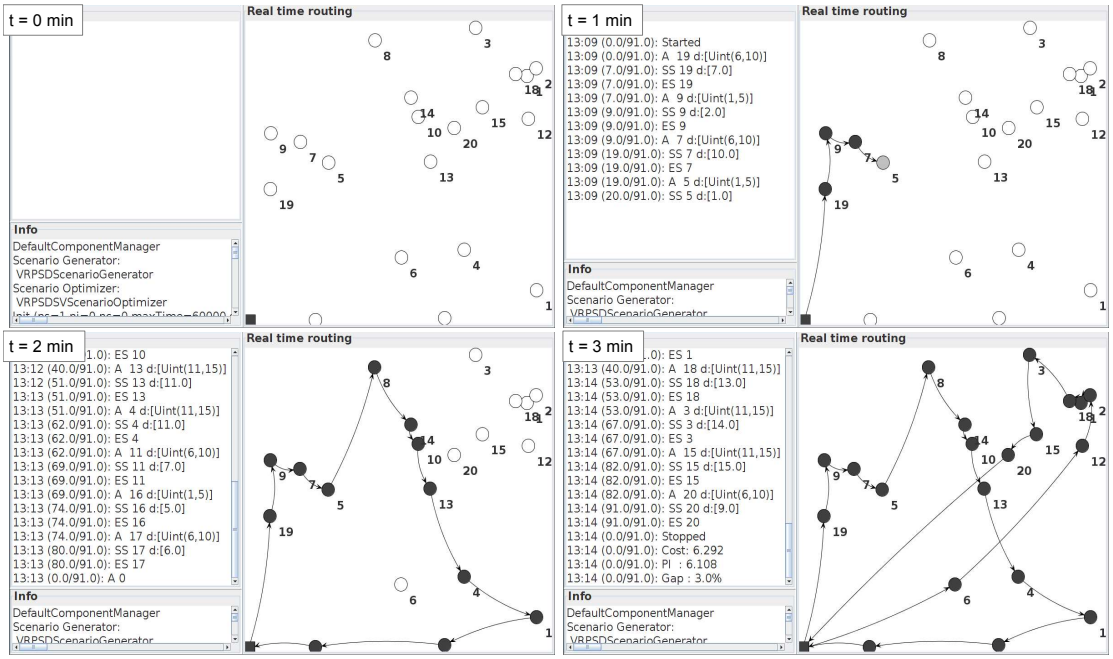

Figure 10: A graphical user interface for jMSA

5 Computational experiments

5.1 Instances

The benchmark instances for the DVRPSD used in this work were initially proposed by Novoa [41] and later used in Novoa and Storer [40]. In this work we consider the larger problems with 30, 40 and 60 customers uniformly distributed in a 1×1 square grid with discrete uniform demands. For each problem size, there are ten combinations of five different demand distributions by two vehicle capacities, leading to a complete testbed of 30 instances.

For each original instance (combination of problem size, demand distribution, and vehicle capacity) we generated 100 possible realizations of the customer demands, creating a complete testbed of 3000 instances. Each instance was solved to optimality using the COIN-OR Symphony VRP solver [49, 50].

5.2 CW+AVNS

To assess the optimization component in isolation we conducted an experiment on the 3000 instances derived from the Novoa [41] benchmark. Fig. 11 presents the distribution of gaps to optimal values for the CW+AVNS algorithm and a CW+2-opt heuristic used as comparison. Note that CW+AVNS clearly dominates CW+2-opt, with 90% of all instances solved with a gap of less than 4%. Additionally, CW+AVNS runs relatively fast, with average CPU times between 50 ms and 650 ms for the larger instances.

5.3 DVRPSD

Our experimental setting is comparable to the one presented in Novoa and Storer [40]. For each instance we ran a simulations using the jMSA framework as a black box. This means that an external simulator was used to send events to the MSA procedure simulating the vehicle route execution.

For easier comparison we report results in terms of *value of information* [37]. The value of information for instance I, namely $\mathcal{V}(I)$, is the gap between the cost of the final solution returned

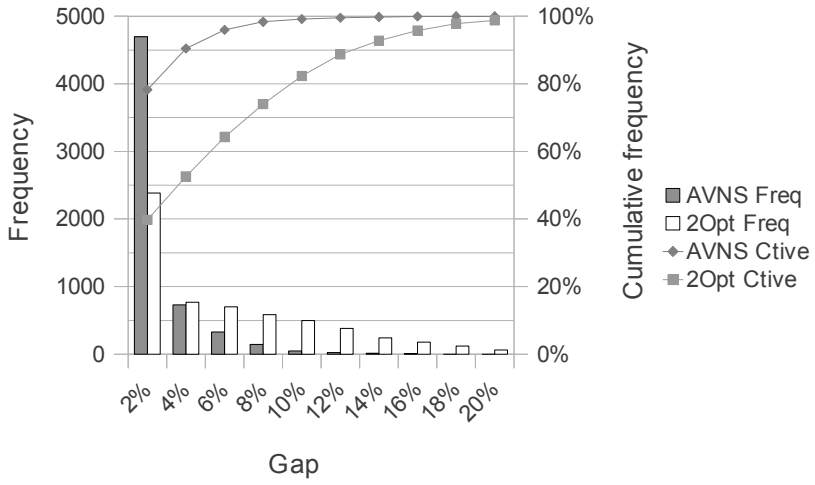

Figure 11: Optimal gap distribution of the CW+AVNS algorithm vs. CW+2-opt for all Novoa [41] instances

by the algorithm $z(I)$ and the a-posteriori optimal solution $z^*(I)$, and it is calculated as follows:

$$\mathcal{V}(I) = \frac{z(I) - z^*(I)}{z^*(I)} \tag{1}$$

	Instance set (size,capacity)						
Algorithm	**(30,137)**	**(30,87)**	**(40,183)**	**(40,116)**	**(60,274)**	**(60,175)**	**Average**
1s_n2_r [51]	12.3%	11.8%	11.1%	12.9%	13.9%	19.6%	13.6%
1s_stostat_r [40]	4.7%	5.1%	3.7%	**5.3%**	3.5%	12.3%	5.8%
2s_stostat_r [40]	3.5%	**3.6%**	**3.0%**	5.4%	**2.8%**	10.7%	4.8%
jMSA	**0.9%**	4.1%	3.5%	6.3%	2.9%	**2.0%**	**3.3%**

Table 2: Comparison of average value of information.

Table 2 presents the results for problems from the Novoa testbed [41]. Because of the aggregated results reported by Novoa and Storer [40], we report the *average value of information* by using *average* solution values in Eq. 1. MSA dominates the algorithm proposed by Secomandi [51] (1s_n2_r), and outperforms the best performing algorithms reported by Novoa and Storer [40] (1s_stostat_r, 2s_stostat_r) for instances with 30 and 60 customers, and a vehicle capacity of 137 and 175. Additionally, MSA shows better overall results with an average gap of 3.3% against 4.8% for 2s_stostat_r, 5.8% for 1s_stostat, and 13.6% for 1s_n2_r. Aside from the performance in terms of value of information, it is important to stress that MSA runs continuously, and the next customer to visit is selected in a fraction of a second, while the other algorithms can take up to several minutes to make such decision, limiting their deployment and applicability in a real-world online DSS.

Aside from direct numerical comparison, the strength of our approach relies on the lack of strong assumptions on demand distributions. To illustrate this point, we adapted the testbed instances by converting demand discrete uniform distributions into left-truncated normal distributions ($\mathcal{N}_{LT\geq 0}$) as follows:

$$\mathcal{U}_{int}\,(a,b) \rightarrow \mathcal{N}_{LT\geq 0}\left(\frac{a+b}{2}, \frac{b-a+2}{6}\right) \tag{2}$$

Note that Eq. 2 ensures that the demand will be between $a-1$ and $b+1$ with probability 0.997, and truncates negative values.

Table 3 highlights the robustness of MSA which shows consistent performance when demand distributions are changed from uniform (discrete) to normal (continuous). Further, the results are

	Instance set (size,capacity)						
Algorithm	**(30,137)**	**(30,87)**	**(40,183)**	**(40,116)**	**(60,274)**	**(60,175)**	**Average**
Uniform	0.9%	3.9%	3.5%	6.3%	2.9%	2.0%	3.3%
Normal	0.7%	3.6%	3.4%	6.2%	2.2%	1.9%	3.0%

Table 3: Comparison of average VI for discrete uniform and normal distributions.

as expected slightly better, with a reduction of 0.3% in the overall average value of information, which is due to the smaller variance. It is important to stress that to conduct this experiment the only change required was to use a different random number generator, which illustrates the flexibility of our approach. Other approaches based on stochastic modeling, like those of Novoa and Storer [40], are not as flexible and heavily depend on distributional assumptions.

6 Conclusions

In this paper we presented the design and implementation of jMSA, an object-oriented event-driven framework for the Multiple Scenario Approach (MSA). By doing a high-level abstraction of MSA to a problem independent level, we modeled it as an event-driven process that allows high reactivity to changes occurring in online and highly dynamic operational environments. We implemented jMSA as a flexible framework that is easily embeddable in decision support systems. By design, jMSA includes a callback system that gives the user further control over MSA and allow complex interactions with third party components. Additionally, we integrated into the framework the parallelization of time consuming tasks with no compromise for the framework user, which is a key aspect considering the wide availability of multi-core personal computers.

We illustrated the use of jMSA on the DVRPSD. The optimization of scenarios is performed by an Adaptive Variable Neighborhood Search (AVNS) which improves an initial solution generated with a randomized Clarke and Wright heuristic. The strength of AVNS is that it automatically adjusts its search scheme depending on the problem's structure by keeping track of the neighborhood performance throughout the execution of the MSA procedure. Computational experiments show that our approach is competitive with state-of-the-art algorithms that take full advantage of the stochastic aspects, while it provides a more flexible scheme that can be used to tackle problems with more general demand distributions.

Acknowledgements

Financial support for this work was provided by the Region Pays de la Loire (France), as part of the Vallée du Libre project, and the Centro de Estudios Interdisciplinarios Básicos y Aplicados en Complejidad (CeiBA, Colombia). This support is gratefully acknowledged.

References

[1] Attanasio, A., Bregman, J., Ghiani, G., and Manni, E. (2007). Real-time fleet management at Ecourier Ltd. In Zeimpekis, V., Tarantilis, C. D., Giaglis, G. M., and Minis, I., editors, *Dynamic Fleet Management*, volume 38 of *Operations Research/Computer Science Interfaces*, chapter 10, pages 219–238. Springer US.

[2] Baldacci, R., Toth, P., and Vigo, D. (2007). Recent advances in vehicle routing exact algorithms. *4OR: A Quarterly Journal of Operations Research*, 5(4):269–298.

[3] Barcelo, J., Grzybowska, H., and Pardo, S. (2007). Vehicle routing and scheduling models, simulation and city logistics. In Zeimpekis, V., Tarantilis, C. D., Giaglis, G. M., and Minis, I., editors, *Dynamic Fleet Management*, volume 38 of *Operations Research/Computer Science Interfaces*, pages 163–195. Springer US.

[4] Beaudry, A., Laporte, G., Melo, T., and Nickel, S. (2010). Dynamic transportation of patients in hospitals. *OR Spectrum*, 32:77–107.

[5] Bent, R. W. and Van Hentenryck, P. (2004a). Regrets only! online stochastic optimization under time constraints. In *Proceedings of the 19th National Conference on Artificial Intelligence (AAAI-04)*, pages 501–506. AAAI Press.

[6] Bent, R. W. and Van Hentenryck, P. (2004b). Scenario-based planning for partially dynamic vehicle routing with stochastic customers. *Operations Research*, 52(6):977–987.

[7] Benyahia, I. and Potvin, J. Y. (1998). Decision support for vehicle dispatching using genetic programming. *IEEE Transactions on Systems Man and Cybernetics Part A - Systems and Humans*, 28(3):306–314.

[8] Bieding, T., Görtz, S., and Klose, A. (2009). On line routing per mobile phone : A case on subsequent deliveries of newspapers. In Beckmann, M., Künzi, H. P., Fandel, G., Trockel, W., Basile, A., Drexl, A., Dawid, H., Inderfurth, K., Kürsten, W., Nunen, J. A., Speranza, M. G., and Bertazzi, L., editors, *Innovations in Distribution Logistics*, volume 619 of *Lecture Notes in Economics and Mathematical Systems*, pages 29–51. Springer Berlin Heidelberg.

[9] Chang, H., Givan, R., and Chong, E. (2000). Online scheduling via sampling. In *Proceedings of the Artificial Intelligence Planning and Scheduling (AIPS 2000)*, pages 62–71.

[10] Clarke, G. and Wright, J. W. (1964). Scheduling of vehicles from a central depot to a number of delivery points. *Operations Research*, 12(4):568–581.

[11] Cordeau, J.-F., Laporte, G., Savelsbergh, M. W., and Vigo, D. (2007). Vehicle routing. In Barnhart, C. and Laporte, G., editors, *Transportation*, volume 14 of *Handbooks in Operations Research and Management Science*, chapter 6, pages 367 – 428. Elsevier.

[12] Crainic, T. G., Gendreau, M., and Potvin, J.-Y. (2009). Intelligent freight-transportation systems: Assessment and the contribution of operations research. *Transportation Research Part C: Emerging Technologies*, 17(6):541–557.

[13] Dantzig, G. and Ramser, J. (1959). The truck dispatching problem. *Management Science*, 6(1):80–91.

[14] Eksioglu, B., Vural, A. V., and Reisman, A. (2009). The vehicle routing problem: A taxonomic review. *Computers & Industrial Engineering*, 57(4):1472 – 1483.

[15] Fleischmann, B., Gnutzmann, S., and Sandvoss, E. (2004). Dynamic vehicle routing based on online traffic information. *Transportation Science*, 38(4):420–433.

[16] Gambardella, L., Rizzoli, A., Oliverio, F., Casagrande, N., Donati, A., Montemanni, R., and Lucibello, E. (2003). Ant colony optimization for vehicle routing in advanced logistics systems. In *Proceedings of the International Workshop on Modelling and Applied Simulation (MAS 2003)*, pages 3–9.

[17] Gendreau, M., Guertin, F., Potvin, J.-Y., and Taillard, E. (1999). Parallel tabu search for real-time vehicle routing and dispatching. *Transportation Science*, 33(4):381–390.

[18] Godfrey, G. and Powell, W. B. (2002). An adaptive dynamic programming algorithm for dynamic fleet management, I: Single period travel times. *Transportation Science*, 36(1):21–39.

[19] Goel, A. and Gruhn, V. (2008). A general vehicle routing problem. *European Journal of Operational Research*, 191(3):650–660.

[20] Haghani, A. and Jung, S. (2005). A dynamic vehicle routing problem with time-dependent travel times. *Computers & Operations Research*, 32(11):2959 – 2986.

[21] Hvattum, L. M., Lokketangen, A., and Laporte, G. (2006). Solving a dynamic and stochastic vehicle routing problem with a sample scenario hedging heuristic. *Transportation Science*, 40(4):421–438.

[22] Hvattum, L. M., Lokketangen, A., and Laporte, G. (2007). A branch-and-regret heuristic for stochastic and dynamic vehicle routing problems. *Networks*, 49(4):330–340.

[23] Ichoua, S., Gendreau, M., and Potvin, J.-Y. (2003). Vehicle dispatching with time-dependent travel times. *European Journal of Operational Research*, 144(2):379 – 396.

[24] Ichoua, S., Gendreau, M., and Potvin, J.-Y. (2006). Exploiting knowledge about future demands for real-time vehicle dispatching. *Transportation Science*, 40(2):211–225.

[25] Ichoua, S., Gendreau, M., and Potvin, J.-Y. (2007). Planned route optimization for real-time vehicle routing. In Zeimpekis, V., Tarantilis, C. D., Giaglis, G. M., and Minis, I., editors, *Dynamic Fleet Management*, volume 38 of *Operations Research/Computer Science Interfaces*, pages 1–18. Springer US.

[26] Irnich, S., Funke, B., and Grünert, T. (2006). Sequential search and its application to vehicle-routing problems. *Computers & Operations Research*, 33(8):2405 – 2429.

[27] Kilby, P., Prosser, P., and Shaw, P. (1998). Dynamic VRPs: a study of scenarios. m APES-06-1998, University of Strathclyde, Glasgow, Scotland.

[28] Krumke, S. O., Rambau, J., and Torres, L. M. (2002). Real-time dispatching of guided and unguided automobile service units with soft time windows. In Mhring, R. and Raman, R., editors, *Proceedings of the 10th European Symposium on Algorithms (ESA 2002)*, volume 2461 of *Lecture Notes in Computer Science*, pages 417–424. Springer Berlin / Heidelberg.

[29] Laporte, G. (2007). What you should know about the vehicle routing problem. *Naval Research Logistics (NRL)*, 54(8):811–819.

[30] Laporte, G. (2009). Fifty years of vehicle routing. *Transportation Science*, 43(4):408–416.

[31] Li, J.-Q., Borenstein, D., and Mirchandani, P. B. (2007). A decision support system for the single-depot vehicle rescheduling problem. *Computers & Operations Research*, 34(4):1008 – 1032.

[32] Li, J.-Q., Mirchandani, P. B., and Borenstein, D. (2009). Real-time vehicle rerouting problems with time windows. *European Journal of Operational Research*, 194(3):711 – 727.

[33] Lorini, S., Potvin, J.-Y., and Zufferey, N. (2011). Online vehicle routing and scheduling with dynamic travel times. *Computers & Operations Research*, 38(7):1086 – 1090.

[34] Mendoza, J. E., Castanier, B., Guéret, C., Medaglia, A. L., and Velasco, N. (2010). A memetic algorithm for the multi-compartment vehicle routing problem with stochastic demands. *Computers & Operations Research*, 37(11):1886–1898.

[35] Mendoza, J. E., Medaglia, A. L., and Velasco, N. (2009). An evolutionary-based decision support system for vehicle routing: The case of a public utility. *Decision Support Systems*, 46(3):730 – 742.

[36] Mes, M., Van der Heijden, M., and Van Harten, A. (2007). Comparison of agent-based scheduling to look-ahead heuristics for real-time transportation problems. *European Journal of Operational Research*, 181(1):59–75.

[37] Mitrović-Minić, S. and Laporte, G. (2004). Waiting strategies for the dynamic pickup and delivery problem with time windows. *Transportation Research Part B: Methodological*, 38(7):635–655.

[38] Mladenovic, N. and Hansen, P. (1997). Variable neighborhood search. *Computers & Operations Research*, 24(11):1097 – 1100.

[39] Montemanni, R., Gambardella, L. M., Rizzoli, A. E., and Donati, A. V. (2005). Ant colony system for a dynamic vehicle routing problem. *Journal of Combinatorial Optimization*, 10(4):327–343.

[40] Novoa, C. and Storer, R. (2009). An approximate dynamic programming approach for the vehicle routing problem with stochastic demands. *European Journal of Operational Research*, 196(2):509–515.

[41] Novoa, C. M. (2005). *Static and dynamic approaches for solving the vehicle routing problem with stochastic demands*. PhD thesis, Lehigh University, Pennsylvania, United States.

[42] Pillac, V., Guéret, C., and Medaglia, A. L. (2010). Dynamic vehicle routing: State of the art and prospects. Technical report, École des Mines de Nantes, France.

[43] Potvin, J. Y., Xu, Y., and Benyahia, I. (2006). Vehicle routing and scheduling with dynamic travel times. *Computers & Operations Research*, 33(4):1129–1137.

[44] Powell, W. B. (1988). A comparative review of alternative algorithms for the dynamic vehicle allocation problem. In Golden, B. and Assad, A., editors, *Vehicle Routing: Methods and Studies*, pages 249–291. North Holland, Amsterdam, The Netehrlands.

[45] Powell, W. B. (2007). *Approximate dynamic programming: solving the curses of dimensionality*, volume 703 of *Wiley Series in Probability and Statistics*. Wiley-Interscience, Hoboken, New Jersey.

[46] Powell, W. B. and Topaloglu, H. (2003). Stochastic programming in transportation and logistics. *Handbooks in Operations Research and Management Science*, 10:555–636.

[47] Powell, W. B. and Topaloglu, H. (2005). Fleet management. In Wallace, S. W. and Ziemba, W., editors, *Applications of Stochastic Programming*, volume 5 of *MPS-SIAM series on Optimization*, chapter 12, pages 185–215. SIAM.

[48] Psaraftis, H. (1980). A dynamic-programming solution to the single vehicle many-to-many immediate request dial-a-ride problem. *Transportation Science*, 14(2):130–154.

[49] Ralphs, T. (2006). Symphony user manual.

[50] Ralphs, T., Kopman, L., Pulleyblank, W., and Trotter, L. (2003). On the capacitated vehicle routing problem. *Mathematical Programming*, 94(2):343–359.

[51] Secomandi, N. (2001). A rollout policy for the vehicle routing problem with stochastic demands. *Operations Research*, 49(5):796–802.

[52] Secomandi, N. and Margot, F. (2009). Reoptimization approaches for the vehicle-routing problem with stochastic demands. *Operations Research*, 57(1):214–230.

[53] Simao, H., Day, J., George, A., Gifford, T., Nienow, J., and Powell, W. B. (2009). An approximate dynamic programming algorithm for large-scale fleet management: A case application. *Transportation Science*, 43(2):178–197.

[54] Tagmouti, M., Gendreau, M., and Potvin, J.-Y. (2011). A dynamic capacitated arc routing problem with time-dependent service costs. *Transportation Research Part C: Emerging Technologies*, 19(1):20 – 28.

[55] Taniguchi, E., Thompson, R., Yamada, T., and van Duin, J., editors (2001). *City Logistics: Network Modelling and Intelligent Transport Systems*. Pergamon.

[56] Toth, P. and Vigo, D., editors (2002). *The vehicle routing problem*, volume 9 of *SIAM Monographs on Discrete Mathematics*. SIAM Philadelphia.

[57] Van Hemert, J. I. and Poutré, J. L. (2004). Dynamic routing problems with fruitful regions: Models and evolutionary computation. In Yao, X., Burke, E., Lozano, J. A., Smith, J., Merelo-Guervs, J. J., Bullinaria, J. A., Rowe, J., Tino, P., Kabn, A., and Schwefel, H.-P., editors, *Parallel Problem Solving from Nature*, volume 3242 of *Lecture Notes in Computer Science*, pages 692–701. Springer Berlin / Heidelberg.

[58] Van Hentenryck, P. and Bent, R. (2006). *Online stochastic combinatorial optimization*. MIT Press.

[59] Yang, J., Jaillet, P., and Mahmassani, H. (2004). Real-time multivehicle truckload pickup and delivery problems. *Transportation Science*, 38(2):135–148.

[60] Zak, J. (2010). Decision support systems in transportation. In Kacprzyk, J., Jain, L. C., Jain, L. C., and Lim, C. P., editors, *Handbook on Decision Making*, volume 4 of *Intelligent Systems Reference Library*, pages 249–294. Springer Berlin Heidelberg.

[61] Zeimpekis, V., Minis, I., Mamassis, K., and Giaglis, G. M. (2007). Dynamic management of a delayed delivery vehicle in a city logistics environment. In Zeimpekis, V., Tarantilis, C. D., Giaglis, G. M., and Minis, I., editors, *Dynamic Fleet Management*, volume 38 of *Operations Research/Computer Science Interfaces Series*, chapter 9, pages 197–217. Springer US.